01 | 认识 React：如何创建你的第一个 React 应用？

2021-05-24 王沛

《React Hooks 核心原理与实战》 课程介绍 >

讲述：王沛
时长 17:10 大小 15.74M

你好，我是王沛。欢迎来到我们的第一节课：认识 React ，并创建你的第一个 React 应用。

在这一讲，我会先带你了解 React 的创新之处，从而理解为什么它能成为最为主流的前端框架。然后再来学习和理解 React 的基本概念，帮助你了解 React 。最后，我会通过一个实战项目，带着你创建你的第一个 React 应用。

React 的颠覆式创新

React 的中文含义是“反应”或“响应”，它描述了 React 这样一个前端框架的核心原理：**当数据发生变化时，UI 能够自动把变化反映出来**。这在 React 当时出现的背景之下，可以说是**一个颠覆式的创新**。

我之所以用“颠覆”这个词，是因为它不仅提供了一个框架，而且彻底改变了前端的开发思路，甚至电脑桌面、手机应用的开发也受到了 React 开发思路的影响。

在 2013 年 React 出现之时，主流的开发 UI 的方式仍然是基于浏览器 DOM 的 API，去精细地控制 DOM 节点的创建、修改和删除。为了保证 UI 上的一致性，我们需要非常小心地处理因各种数据的变化而导致的 UI 的变化。

举个例子。对于一个聊天应用，当来了一条新消息时，我们一方面需要在聊天框内添加一条新消息，同时也要在显示消息数量的地方让数字加 1，这样才能保证 UI 的一致性。

在 React 之前，我们需要调用 DOM 的 API 来修改 DOM 树的结构，从而改变 UI 的展现。而在有了 React 之后，我们只需要在业务状态和 UI 状态之间建立一个绑定的关系就行了。绑定完成后，我们就不需要再关心怎么去精细控制 UI 的变化，因为 React 会在数据发生变化时，帮助我们完成 UI 的变化。

下面这张图就展示了这样的一个机制：

可以看到，我们可以通过 JSX 语法，用声明式的方式来描述数据和 UI 之间的关系，那么数据在发生变化时，UI 也会自动发生变化。这样的话，无论是收到一条还是多条消息，React 都会自动完成 UI 的展现，我们也就不再需要去关心怎么产生变化的细节。那么基于同一个数据，比如我们需要在通知栏里显示消息的数量，那么显示消息数量的组件，只需要绑定到消息的长度上，它也会自动更新，这样很容易就保证 UI 上的一致性了。

通过我刚才对这张图的解释，你可能还会有些不理解，主要是出现了一些比较陌生的概念。别着急，接下来我就带你理解 React 的基本概念，学完之后，你会对 React 是如何工作的有更全面的认识。

理解 React 的基本概念

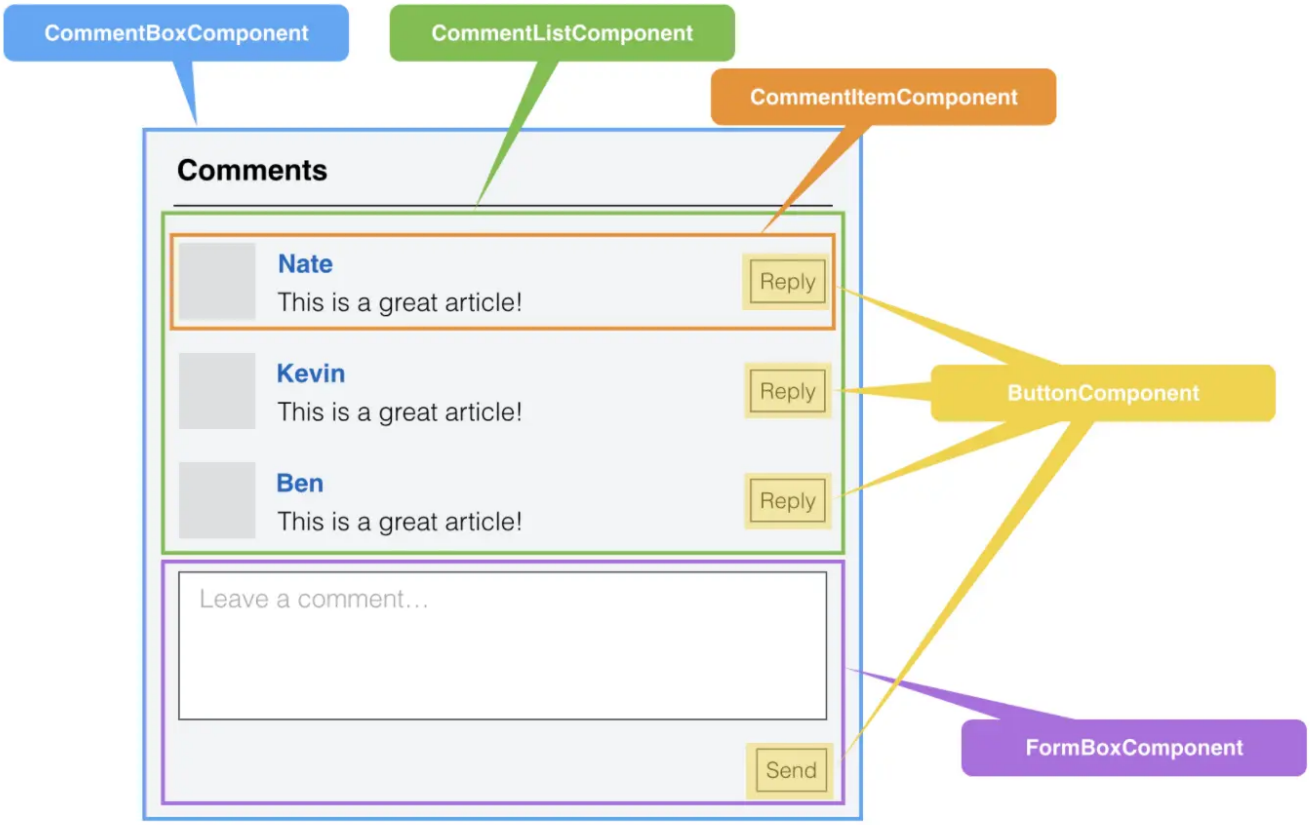
React 本身其实是一个非常简单的框架，要理解它的用法，无外乎就是理解下面三个概念：组件、状态和 JSX。下面我们分别来看。

使用组件的方式描述 UI

在 React 中，所有的 UI 都是通过**组件**去描述和组织的。可以认为，React 中所有的元素都是组件，具体而言分为两种。

1. 内置组件。内置组件其实就是**映射到 HTML 节点的组件**，例如 div、input、table 等等，作为一种约定，它们都是小写字母。
2. 自定义组件。自定义组件其实就是**自己创建的组件**，使用时必须以大写字母开头，例如 TopicList、TopicDetail。

和 DOM 的节点定义类似，React 组件是以**树状结构**组织到一起的，一个 React 的应用通常只有一个根组件。此外，在实际开发时，我们会把 UI 划分成不同的组件，然后组织到一起，例如对于下面这样一个评论框的 UI：



我们可以用基于组件的方式去描述：

复制代码

```
function CommentBox() {
  return (
    <div>
      <CommentHeader />
      <CommentList />
      <CommentForm />
    </div>
  );
}
```

比如说评论框可以分为三个部分，包括头部、评论的列表和一个用于提交新的评论的表单，这样我们就可以把这三个部分分别定义成组件，让我们可以根据实际的场景把复杂的 UI 模块化为独立的组件。这样代码不仅看起来更加直观，而且也更容易维护。

使用 state 和 props 管理状态

正如我刚才提到的，React 的核心机制是能够在数据发生变化的时候自动重新渲染 UI，那么势必要有一个让我们保存状态的地方，这个保存状态的机制就是 state。而 props 就是类似于 Html 标记上属性的概念，是为了在父子组件之间传递状态。

在函数组件中，我们可以使用 useState 这样一个 Hook 来保存状态，那么状态在发生变化时，也会让 UI 自动发生变化。比如下面这段代码，展示了一个简单计数器的实现的例子：

复制代码

```
import React from "react";

export default function Counter() {
  const [count, setCount] = React.useState(0);
  return (
    <div>
      <button onClick={() => setCount(count + 1)}>{count}</button>
    </div>
  );
}
```

可以看到，通过 useState 定义这样一个状态，让这个状态来保持计数器的数值，那么在值发生变化时，组件就会自动重新刷新。这里我们先不用太关心 useState 这个 API 的细节，我在第 4 讲会有一个详细的介绍。

那么，现在使用了 state 来维护组件的状态，接下来要关心的就是组件之间的交互，这正是 props 提供的作用。

无论是 div、span 这样的内置组件，还是自定义组件，都可以在使用时把接收属性作为参数。而当这个参数发生变化时，组件也就会自动重新渲染。

例如，我们在计数器这个例子中使用一个组件来渲染 count 这个值，要求在值大于 10 的时候显示为红色，否则就为蓝色：

复制代码

```
import React from "react";

function CountLabel({ count }) {
  // 子组件用于显示颜色
  const color = count > 10 ? "red" : "blue";
  return <span style={{ color }}>{count}</span>;
}

export default function Counter() {
  // 定义了 count 这个 state
  const [count, setCount] = React.useState(0);

  return (
    <div>
      <button onClick={() => setCount(count + 1)}>
        <CountLabel count={count} />
      </button>
    </div>
  );
}
```

可以看到，我们定义了一个新的组件 CountLabel，在值大于 10 的时候显示红色，否则为蓝色。并且，我们还要在 Counter 组件里使用 CountLabel 这个子组件，这样的话我们就可以通过 props 把 count 这个值从父组件传递到子组件，那么在 count 发生变化时，CountLabel 也会重新渲染。此外，所有通过属性定义在这个 Tag 上的参数，都会作为一个对象传递给函数组件，这样在函数组件内部就可以使用这些参数了。

理解 JSX 语法的本质

我们刚刚通过一个简单的例子对 React 组件有了一个比较直观的印象，但如果你之前没有接触过 React，也许会觉得在代码中同时包含 JavaScript 和 HTML 标记的写法很别扭。但

这种写法其实正是 React 中的“模板语言”：JSX。

注意，这里的“模板语言”是加了引号的，因为从本质上来说，JSX 并不是一个新的模板语言，而可以认为是一个**语法糖**。也就是说，不用 JSX 的写法，其实也是能够写 React 的。这是什么意思呢？我们不妨再看一下上面计数器的例子，如果不用 JSX 应该如何写代码：

复制代码

```
React.createElement(
  "div",
  null,
  React.createElement(
    "button",
    { onClick: function onClick() {
        return setCount(count + 1);
      } },
    React.createElement(CountLabel, { count: count })
  )
);
```

在这段代码中，JSX 的部分我们是用 JavaScript 的方式去实现的，并且用到了 React.createElement 这样一个 API，它的作用就是创建一个组件的实例。此外，这个 API 会接收一组参数：

第一个参数表示组件的类型；

第二个参数是传给组件的属性，也就是 props；

第三个以及后续所有的参数则是子组件。

所以呢，通过 createElement 这个 API，我们可以构建出需要的组件树，而 JSX 只是让这种描述变得更加直观和高效。所以我们说 JSX 其实是一种语法糖。理解这一点非常重要，因为它意味着两点：

1. JSX 的表达能力等价于 JavaScript 的表达能力，那么所有我们可能需要的机制，比如循环、条件语句等等，JSX 其实都能灵活表达。
2. JSX 几乎不需要学习，只要你会用 JavaScript，就也会用 JSX。

所以这也是 React 的“模板语言”区别于 Angluar 和 Vue 的地方，JSX 不是一个新的概念，而只是原生 JavaScript 的另一种写法。但是换成这种写法，就会大大降低你上手 React 的难度。

使用脚手架工具创建 React 应用

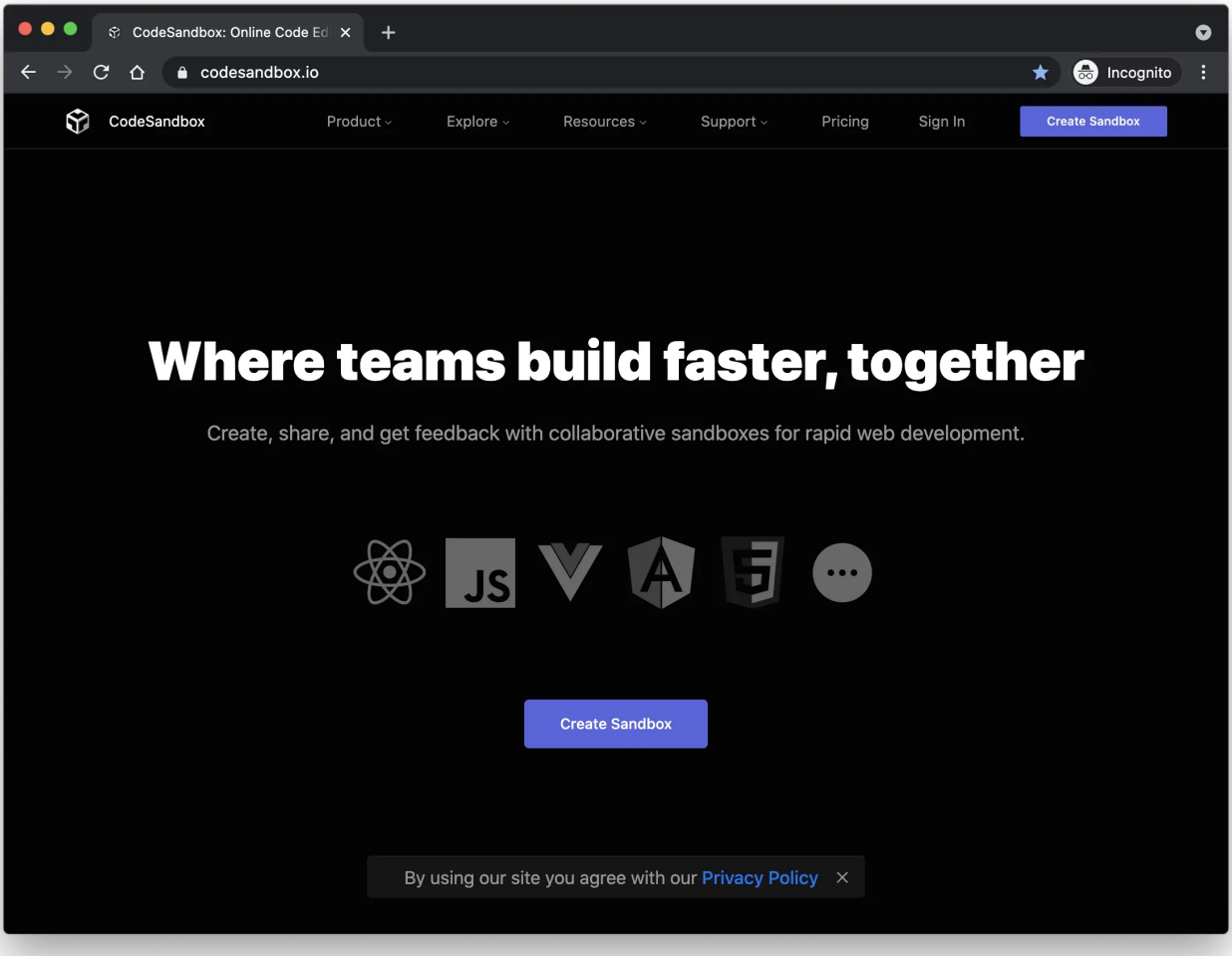
经过刚才的学习，我们已经对 React 有了一个基本的了解，那么现在我们就来创建一个应用，试用一下 React。

在实际的项目开发中，我们其实不仅需要把 React 作为 UI 层，还可能需要路由管理、状态管理等框架。与此同时，我们还需要使用最新的 JavaScript 语言特性、CSS 的预处理框架等等。所以一般需要结合 Webpack 等打包工具来使用 React。

完成这些配置本身就是一个比较繁琐的步骤，因此我们通常都不会从头开始创建一个项目，而是使用**脚手架工具**来创建一个项目。我在这里跟你介绍两个工具：

> 一个是 codesandbox.io，主要用于学习 React，能够快速试验一些 React 的特性。
>
> 另一个则是 Facebook 官方提供的 create-react-app 命令行工具，用来创建一个基础的 React 项目。

首先看 Codesandbox，它是一个在线的工具，可以通过 https://codesandbox.io 访问使用。例如今天这节课所有可运行的示例代码就都是放在 Codesandbox 上的，用浏览器打开即可。你可以在线写代码，并实时查看运行的效果。使用起来其实也非常简单，打开主页后就可以看到如下页面：



点击 Create Sandbox，就可以创建一个前端的应用。需要说明的是，它不仅支持 React，还能够创建 Angular、Vue 等前端框架的应用。

第二个 create-react-app 则是我们做正式项目开发时会使用的基本工具。它会提供一个完善的 Webpack 配置，让我们能够立刻开始使用 React、JavaScript 语言的最新特性和 CSS Module 等主流的技术方案。

这里要特别注意一点，create-react-app 的使用需要 Node.js 的环境，具体版本是 Node >= 10.16 版本以及 npm >=5.6 版本。安装好 Node.js 之后，你就可以用如下命令行来创建一个 React 应用：

复制代码

```
npx create-react-app my-app
cd my-app
npm start
```

创建的项目结构非常简单：

```
- public
- src
  - App.css
  - App.js
  - index.js
  - index.css
- package.json
```

这里的 src/index.js 是程序的入口，App.js 则是整个应用程序的根组件，主要做一些配置的事情。create-react-app 创建的项目作为一个基本的模板，为我们提供的便利主要是配置好开发环境、Webpack 等等，但是它不包含 Redux 或者 Router 等非必须的功能。那么基于这个模板，我们后面的几节课会在 App.js 中添加 Redux、React Router 等配置。

实战：在组件中发送请求并显示数据

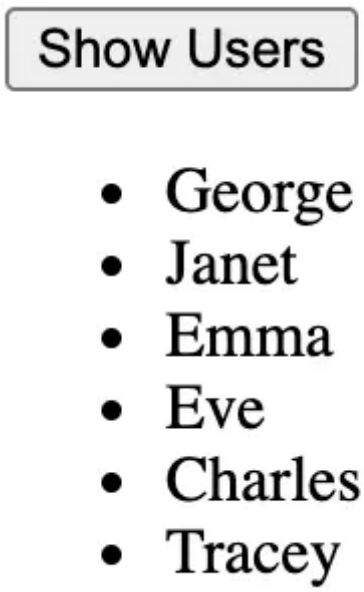
好了，我们已经对 React 有了一个大概的了解，并能够创建项目，试验代码执行的效果。现在我们来通过一个异步请求的例子，以强化对 React 的理解，这个例子也是日常开发中最为常见的需求。

设想这样一个场景：页面上有一个按钮，点击后，我们可以发起一个请求获取一个用户列表，并要求显示在页面上。在这个过程中，我们需要考虑数据状态，Loading 的状态，以及请求出错的处理。那么可以用下面的代码实现：

```
import React from "react";

export default function UserList() {
  // 使用三个 state 分别保存用户列表，loading 状态和错误状态
  const [users, setUsers] = React.useState([]);
  const [loading, setLoading] = React.useState(false);
  const [error, setError] = React.useState(null);

  // 定义获取用户的回调函数
  const fetchUsers = async () => {
    setLoading(true);
    try {
      const res = await fetch("https://reqres.in/api/users/");
      const json = await res.json();
```

```
      // 请求成功后将用户数据放入 state
      setUsers(json.data);
    } catch (err) {
      // 请求失败将错误状态放入 state
      setError(err);
    }
    setLoading(false);
  };

  return (
    <div className="user-list">
      <button onClick={fetchUsers} disabled={loading}>
        {loading ? "Loading..." : "Show Users"}
      </button>
      {error &&
        <div style={{ color: "red" }}>Failed: {String(error)}</div>
      }
      <br />
      <ul>
        {users.length > 0 &&
          users.map((user) => {
            return <li key={user.id}>{user.first_name}</li>;
          })}
      </ul>
    </div>
  );
}

```

程序的运行结果如下图所示：



当点击 Show Users 按钮时，会显示 Loading 的文本，并将 button 设为 disabled。请求成功后，则显示获取的用户列表。如果请求失败，则显示一段错误信息的文本。

在 React 组件中，任何一个 state 发生变化时，整个函数组件其实是被完全执行一遍的，而且除了 state，多次的执行之间没有任何关系。所以在考虑这样一个场景的实现时，我们的思考方式就是要首先考虑这个组件有哪些状态（state），这些状态的变化是由什么触发的，从而将整个功能串联起来。

当然，这个例子只是一个最简单的实现，但如果你细究一下，应该还会提出下面的问题：

1. 函数中定义了回调函数 fetchUsers，但函数每次都是全部重新执行，那会不会重复定义很多次呢？
2. 如果另外一个组件可能也需要使用到 Users 这个数据，比如一个下拉框，那么是不是每次都要重复这个发起请求的逻辑呢？

这其实正是后面的课程会解决的问题，我在这里先简单回答下：

1. 是的，这种写法会重复定义很多函数。不过为了避免这样的问题，React 提供了 useCallback 这样一个 Hook 来缓存回调函数，关于这一问题，我在第 4 讲会有详细的讲解。
2. 对于异步请求逻辑的重用，这其实也意味着跨组件状态的重用，我会在第 7 讲利用 Redux 这样一个全局状态管理框架来实现异步逻辑的复用。

小结

React 本身其实是一个上手非常简单的 UI 框架，它的核心 API 在这节课我们基本都已经用到了，比如函数组件，JSX，useState，等等。但是 React 本身毕竟只是解决了 UI 的问题，在真正开发时，我们还需要路由、需要状态管理等等，这也正是后续课程中会介绍的。

不过也正是因为 React 本身的这种纯粹和简单，让它的整个生态圈变得非常繁荣。针对每一个具体的问题，可能都存在特定的解决方案。所以在实际使用中，我们需要根据具体的需求，来寻求最合适的方案。

最后，我把这节课用到的例子链接放在这里，供你参考。

计数器的例子：https://codesandbox.io/s/admiring-christian-20vzg?file=/src/01/Counter.js

异步请求的例子：https://codesandbox.io/s/admiring-christian-20vzg?file=/src/01/UserList.js

思考题

作为第一讲的思考题，我先来提一个开放式的问题。在你看来，React 最打动你的特性是什么？或者说你认为它的最大优点有哪些？

欢迎在留言区写下你的思考和想法，我们一起交流讨论。如果今天的内容让你有所收获，也欢迎你把课程分享给你的同事、朋友，我们一起共同进步！

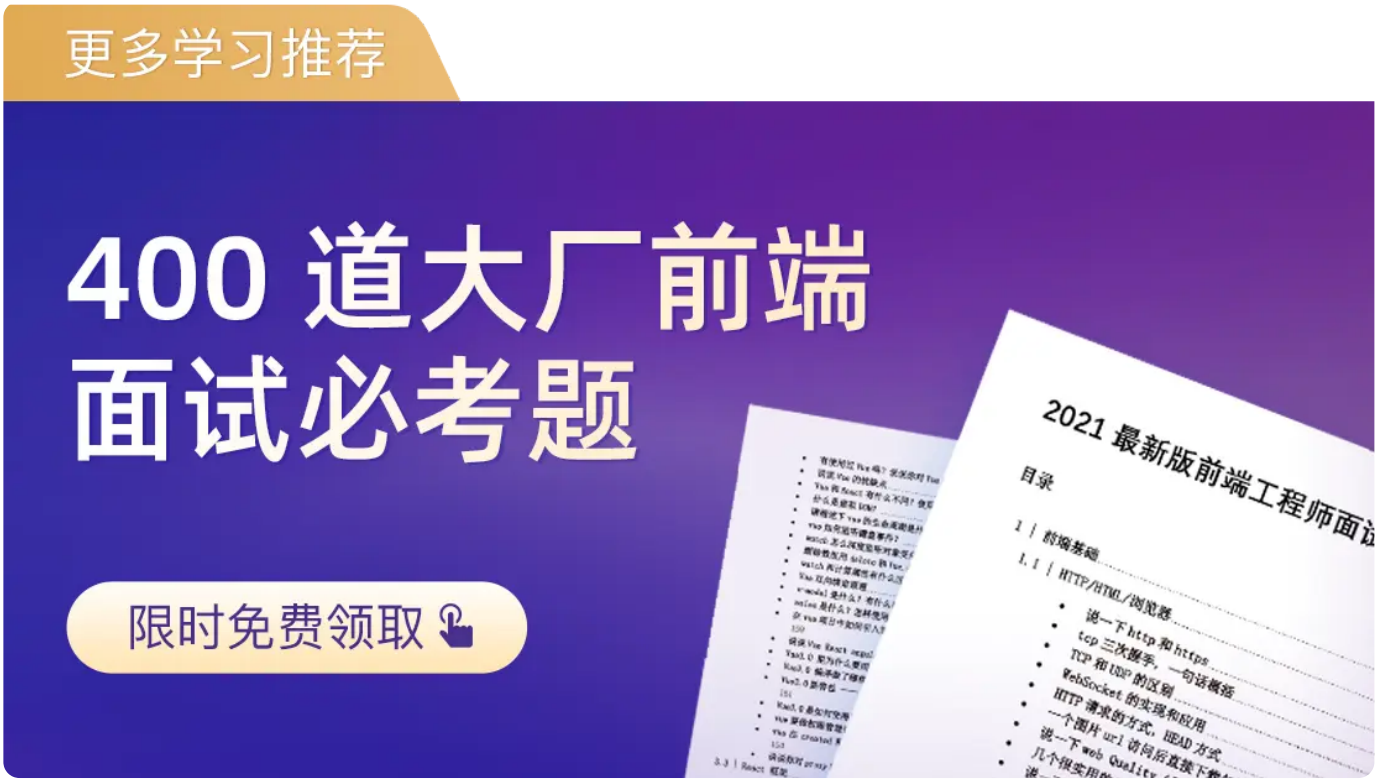
分享给需要的人，Ta订阅后你可得 20 元现金奖励

赞 15 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 开篇词 | 全面拥抱 Hooks，掌握最新 React 开发方式

下一篇 02 | 理解 Hooks：React 为什么要发明 Hooks？



精选留言 (27)

写留言

程 置顶
2021-05-24

应该是jsx吧，灵活，随心所欲。但是就是太灵活，缺少一些最佳实践

> 作者回复: JSX 确实是 React 最亮眼的创新之一，似模板语言而本质是 JavaScript，所以特别灵活是正常的。JSX 写的太长确实容易看着比较凌乱，所以要尽量把组件进行拆分成子组件各司其职。JSX 也就自然看着简洁了。

7

sugar 置顶
2021-05-27

我大概从2015年左右开始关注过这个前端框架，对我来说react比较打动人的地方主要有3点：
1 jsx 的业务代码形态 使得xml 和 js很好地结合，低成本复用很多js的生态
2 在当时 react给了一批较为资深的jser一种“希望”，能够让自己写的业务逻辑 脱离dom，为什么要脱离dom呢？因为2015年正是移动端开发较为火热的时候，react的这种...
展开

作者回复: 说的很好~ JSX 同时满足了模板语言的直观和 JS 的灵活性。虚拟 DOM 的存在让业务逻辑和展现逻辑的分离可以更彻底，拆分了复杂度也更容易去测试。至于 FP，并不是刻意追求的目标，也和 React 没有太大的关系。更多算是一种巧合吧，因为 immutable 的缘故，很多就看上去很像了。

3

刘大夫
2021-05-25

React让我感觉最惊艳的地方在于它的设计哲学，以及对于技术场景不断地深入探索，尤其fiber这一点太厉害了。在这门课中希望老师多讲点 hooks 原理的知识，最好能深入点，正课中可能照顾到大多数读者还是需要以应用实践为主，但可不可以用加餐或者答疑的形式来啊，万分感谢

作者回复: 好的，谢谢提议，有些内容确实可以通过加餐形式去补充。课程中的思路主要是面向实战，因此各个知识点主要讲清楚来龙去脉和要解决的问题。

9

Geek_9878c1
2021-05-25

React初看复杂，真的上手了就会发现就是三板斧jsx，状态管理，组件；三板斧能学完的，第一步就成功了

作者回复: 没错。技术上很容易上手，然后就是面对复杂问题时寻求最优解了 ☺

4

Aaron
2021-05-25

生态强大可靠，应用广泛
核心概念简单，写法灵活
通过RN可开发移动app

展开

作者回复: 回答的很到位！React 核心功能的完备和至简让其上手很简单，同时也促成了整个生态的形成。所以可以说简单和生态是 React 最强力的武器。

4

ru
2021-06-10

如果说最，我觉得是数据驱动页面渲染，减少dom的操作，其次是jsx语法，页面整体逻辑看起来会很清晰

作者回复: 👍

2

qiutian00
2021-06-08

老师， 请教在项目需求多变情况下，怎样进行单元测试之类，来保证项目的质量？

作者回复: 第18课会讲测试哦~

1 2

Jerryz
2021-06-07

React 哲学：
UI = render(state)

作者回复: 👍

2

Ran
2021-05-25

老师，日常遇到的选型问题，像现在大一些的公司都是偏向React技术栈，小规模团队快速开发好像更倾向于Vue，具体项目的技术选型有更多其他的因素吗？

展开

作者回复: 框架确实没有绝对的优劣，更多的是技术团队自身的技术积累决定的。通常来说生态更加繁荣的会更有生命力，选它就对了 ☺

2 2

傻子来了快跑、
2021-05-24

老师，能更新快点嘛

展开

作者回复: 保证每周至少三篇哈，能快会尽量快 😜

1 2

咚门
2021-07-03

当然是jsx了，直接让公用ui库写起来通俗易懂。

展开

1

琼斯基亚
2021-05-25

王老师什么时候Rekit也支持Hooks？

展开

作者回复: 一年前就支持啦~

1

Geek_12f953
2021-07-18

从vue转过来，觉得自定义hooks提取逻辑复用是我比较期待的

作者回复: 👍

重生
2021-06-17

function is No1

展开

健牌哥.
2021-06-14

hook特性可以抽离绝大部分逻辑，让用户在大多数时间可以忘记HOC这个晦涩的用法；搭配jsx让代码逻辑变得更简洁直观。

展开

陶佳佳
2021-06-05

老师，进行异步请求的例子，出现跨域问题， 请问如何解决??

> 作者回复: 跨域是需要服务器端提供 Access-Control-Allow-Origin 这个 header 来允许跨域请求的。

守望
2021-06-04

我怎么觉得react会用的和不会用写出来的是一个天一个地

展开

寇云
2021-06-04

编程风格思想和繁荣的生态圈：
接触到Remotion(Write videos in React) /React 3D/ react-pdf 这样的库之后是用react组件的编程风格和思想去构建各种应用。当react-dom 可以被替换，让软件开发的方式有了很大的想象空间。
随着 TS + react的成熟，也让组件的使用者无需关注组件的内部实现，通过熟悉接口。...

展开

> 作者回复: 赞！React 的颠覆式创新，和专注于 View 的方案优化，带来了繁荣的生态圈~

William
2021-06-03

React 简洁高效！

展开

作者回复: 哈哈，咱要说点具体的~

小馋猫

2021-06-01

JSX的本质是使用React.createElement方法创建组件实例，参数分别为组件名称，组件的属性，子组件